Physics-guided Neural Feedforward Control of a High-precision Positioning System

Konstantin Tsonev
Department of Electrical Engineering
Eindhoven University of Technology
Eindhoven, Netherlands
, Sudent ID 1808303
k.tsonev@student.tue.nl

***Abstract*—Feedforward control compensates for disturbances and changes in the reference input, before the output deviates from the setpoint. However, it is prone error, when the model of the disturbances is not known exactly, especially if it is a difficult-to-determine nonlinearity. This paper proposes a Physics-guided Neural Network (PGNN) model as a solution to this problem and investigates its effects on the performance of an industrial high-precision positioning system, when applied in a feedforward configuration. The approach is tested on a simulation of the physical setup in Simulink and the results, as well as future work, are discussed.**

I. Introduction

In the chip industry, demand for accuracy, precision and efficiency is rapidly increasing. Wafer positioning systems are no exception to this, as they are responsible for placing wafer slabs at a specific location with as little offset as possible. However, in practice, this is very difficult to achieve, as there are many parasitic disturbances affecting the performance of the system, which is already required to be immensely precise (on the order of a few microns). The cause for these disturbances may range from limited approximation capabilities of physics–based models [1] to manufacturing tolerances and electromagnetic disturbances [2], [3]. And these parasitic effects are often difficult or impossible to model, due to inherent unpredictability and non-linearity.

To deal with such phenomena, black box Neural Networks (NNs) can be implemented in feedforward control, as they capture complex non-linearities [4] [5]. They can be trained offline and can then be used to estimate the output of the system. Depending on the training data used and the model of the NN, the nonlinear disturbances can be compensated incredibly effectively.

However, NNs require tedious hyper-parameter tuning, lack physical interpretation, and are inherent to poor extrapolation outside the training data set. A fitting approach is to combine a NN with a known physical model, thus creating a Physics-Guided Neural Network (PGNN) [6] [7]. With the utilization of a regularized cost function during training, the poor extrapolation of the NN can be penalized [8]. Thus, divergence of the physics layer from previously identified physical parameters will be minimized, achieving a satisfactory balance between data-fit and physics-based extrapolation [9].

Another problem in feedforward control is that in order for the controller to ensure optimal performance, its model must be equal to the inverse of the model of the plant [10]. In practice, however, the plant model may not be known, may not be estimated exactly or may not be invertible. The PGNN solves this problem, as it maps inputs to specific outputs, based on training, thus circumventing the need for a precise estimate of the inverse of the plant model.

In this paper we apply the PGNN to a rotary motor and see how it affects the performance of the motor. Moreover, The effectiveness of the NN, when it comes to compensating non-linear disturbances, is also evaluated. Finally, the model is validated on both simulations and a real-life setup.

II. Problem statement

In this section the physical setup is shown in Fig.1, the simulation model of the setup is shown in Fig.2 and the problem is formulated.

The feedforward component of the model can be split into two parts - a physics guided part and a nonlinear position-dependent stiffness component. The output of the feedforward can then be represented as

$$Y_{FF} = F_{physics} + F_{stiffness} \tag{1}$$

where $F_{physics}$ is the force, generated from the physics part of the feedforward component and $F_{stiffness}$ is the force, generated from the nonlinear component. $F_{physics}$ can be written as

Fig. 1: Positioning system

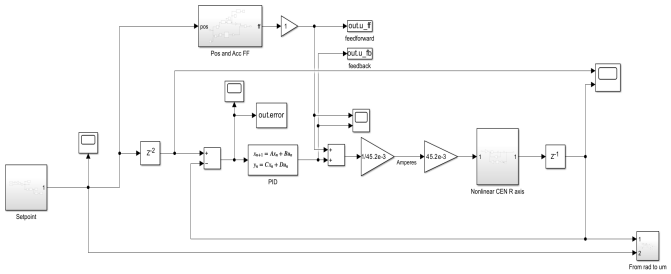

Fig. 2: Simulink model of the physical setup

$$F_{physics} = m.a = m.\frac{d^2}{dt^2}x \tag{2}$$

where m is the mass of the plant and a is the acceleration and x is the input reference position. Furthermore, $F_{stiffness}$ can be written as

$$F_{stiffness} = k(x).x \tag{3}$$

where $k(x)$ describes the nonlinear stiffness relation between the input position and the output force. Since $k(x)$ is nonlinear, as well as position dependent, it is impractical to obtain an analytical expression for this relation.

A. Regression neural networks

A regression neural network (RNN) is a statistical model, which maps a certain input feature to a target output, the goal being to predict a continuous numerical value. By utilizing the vast computing power of a graphics card or a central processing unit, the neural network can produce accurate results very quickly.

Fundamentally, the NN can be viewed as a layered structure, with each layer having a specific number of inputs. The layer inputs are then passed through the following linear function

$$f(x) = w.x + b \tag{4}$$

where w and b are parameters of the NN, also called weights, x being the input and $f(x)$ being the output. After the inputs have been

$$s_j = \sum_{i=1}^{n} w_{ji}.x_i + b_j \tag{5}$$

where s_j is the output of the $j-th$ neuron and n is the number of inputs of the layer, in which the neuron is.

Additionally, the outputs of the neurons are passed through activation functions, in order to introduce nonlinearity into the NN model.

B. Training of a NN

When provided with a dataset, consisting of example values for the input, a second dataset, consisting of desired output values and applying the Gradient Descent method, the neural network can adjust its parameters, so that its predicted output matches the desired output as closely as possible.

To achieve this, a loss function is implemented, which penalizes deviations between the neural network output and the actual output value. For a RNN, usually a mean squared error (MSE) loss function is used

$$J(w,b) = \frac{1}{N}\sum_{i=1}^{N}(f_{w,b}(x^{(i)}) - y^{(i)})^2 \tag{6}$$

where $J(w,b)$ is the normalized cost function, $f_{w,b}(x^{(i)})$ is the predicted output of the NN and $y^{(i)}$ is the desired output. After the cost function is computed, the weights can be adjusted as follows:

$$w = w - \alpha\frac{\partial}{\partial w}J(w,b) \tag{7}$$

$$b = b - \alpha\frac{\partial}{\partial b}J(w,b) \tag{8}$$

where α is a training parameter, called the learning rate, whose value is determined manually. To train the neural network, this process is repeated over hundreds or thousands of iterations, until the RNN converges to a global minimum of $J(w,b)$.

C. Implementation

In Fig.3 the model of the feedforward component is shown. As referenced in eq.(1), its output is the sum of two forces. $F_{stiffness}$ is a force, that is nonlinearly related to position. The nonlinear stiffness relation is approximated via the 1-D T(u) lookup table block, which maps a specific force output to a certain position input.

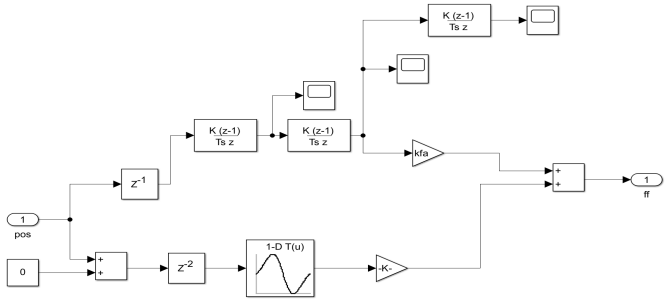

Fig. 3: Feedforward model in Simulink

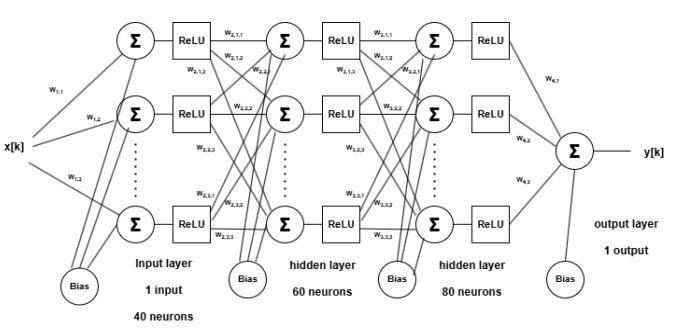

Fig. 4: Architecture of the NN learning the stiffness relation

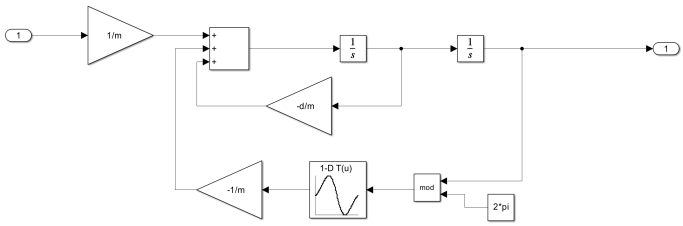

Fig. 5: Plant model in Simulink

To replicate this relation, a NN with architecture shown in Fig.4 is applied. Note that the used ReLU activation function has the following property

$$Relu(z) = max(0, z) \tag{9}$$

where z is the input, which is passed to the ReLU function.

The data used for training the NN has been normalized by implementing the min-max normalization in eq.(1)

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}(u - l) + l \tag{10}$$

where x' is the normalized data, x is the original data, x_{min} and x_{max} are the minimum and maximum values of the original data respectively, u and l are the upper and lower bounds of the new range of the normalized data. So $x' \in [l; u]$.

The following parameters were found to work well in conjunction with the chosen architecture:

$$Learning\ rate = \alpha = 2,7.10^{-3}$$
$$Epochs = 10001$$
$$Batches = 12$$

Moreover, the plant model in Simulink is shown in Fig.5 and its transfer function is determined to be

$$\frac{U(s)}{m} - sY(s).\frac{d}{m} - Y(s)\frac{k(x)}{m} = s^2Y(s) \tag{11}$$

where $U(s)$ is the input of the plant, d is the damping in the plant, s is the Laplace variable and $Y(s)$ is the output of the plant, which is position. Since integration is equivalent to division by s in the s-domain and the output of the plant is position, then eq.(10) can be rewritten as

$$\frac{u}{m} - v.\frac{d}{m} - \frac{k(x).x}{m} = a \tag{12}$$

where u is the result after applying the inverse Laplace transform to $U(s)$ and v is the velocity.

In the case that the system output tracks the reference input perfectly, the error between the two quantities would be zero, hence the output of the PID controller would also be zero. Then

$$u = Y_{FF} = F_{physics} + F_{stiffness} = m.a + k(x).x \tag{13}$$

Substituting eq.(13) into eq.(12) yields

$$v.\frac{d}{m} = 0 \tag{14}$$

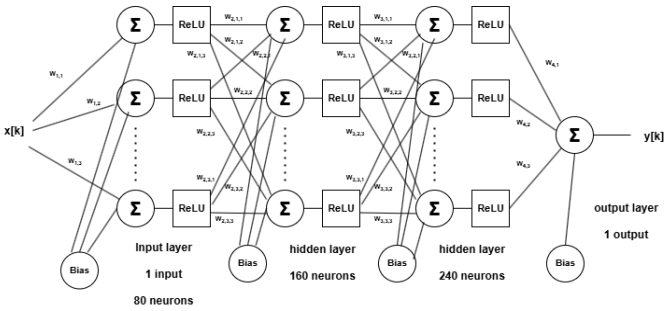

Fig. 6: Architecture of the NN learning the nonlinearity

This suggests that there should also be a damping component present in the feedforward component as well, so that the inverse of the plant is better approximated. Thus, another neural network is implemented in the feedforward to account for the missing damping component. Its architecture is shown in Fig.6 and the parameters used for its training were

$$Learning\ rate = \alpha = 2,7.10^{-3}$$
$$Epochs = 751$$
$$Batches = 12$$

III. Results

A. Neural Network predicting the lookup table block

The NN tracks the output of the 1-D lookup table block very accurately, achieving a mean loss of $6,45.10^{-5}$ across an input dataset with 600 samples. The result is shown in Fig.7.

However, when it comes to the error between the output of the system and the reference input, there is no difference in performance when using the 1-D Lookup table block or when substituting it with a NN. In fact, the output error does not go below 11 um in either scenario.

B. Neural Network predicting the missing damping component

The neural network model, which predicts the missing damping component, achieves a mean loss of $3,45.10^{-7}$ across an input dataset with 10000 samples. The result is shown in Fig.9. The missing damping component is approximated very accurately, with a maximum error of 0,11.

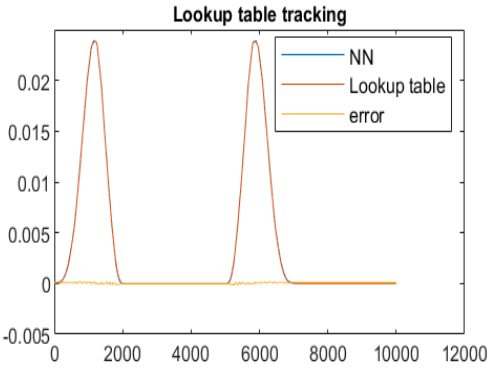

Fig. 7: Lookup table tracking

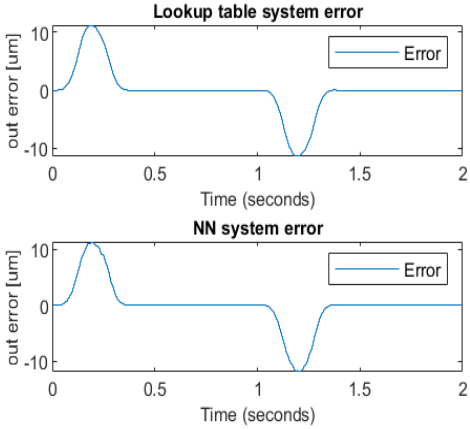

Fig. 8: Error comparison between 1-D block and NN

Now, the error between the output of the system and the reference input has improved slightly. It has decreased by approximately 1 um, compared to the output error, when the 1-D lookup table block is used. The result is shown in Fig.10.

IV. Discussion

When the 1-D lookup table block is replaced by a NN model, the system output error does not change. The reason for this is that the feedforward does not perfectly estimate the inverse of the plant. This results in a nonlinear discrepancy, which affects the system output and is also not related to the 1-D block. So its substitution does not reflect the required change in the feedforward. However, it is proof that neural network models are very effective at learning nonlinear relations and replicating outputs just from inputs.

With this insight obtained, the next logical step is to alter the feedforward component structure, such that its transfer function would match the the inverse of the plant perfectly. By estimating the damping component from the velocity in the feedforward, the error decreases slightly, likely because the nonlinear discrepancy is captured only partially and not entirely.

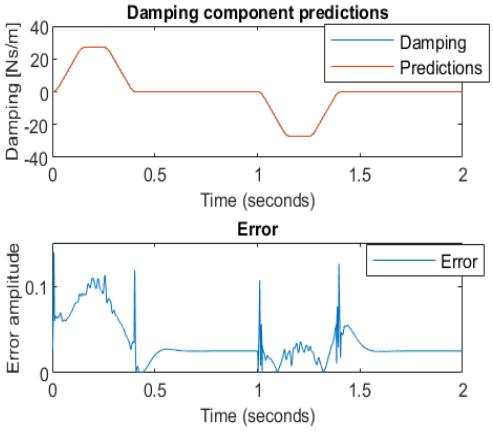

Fig. 9: Damping component tracking

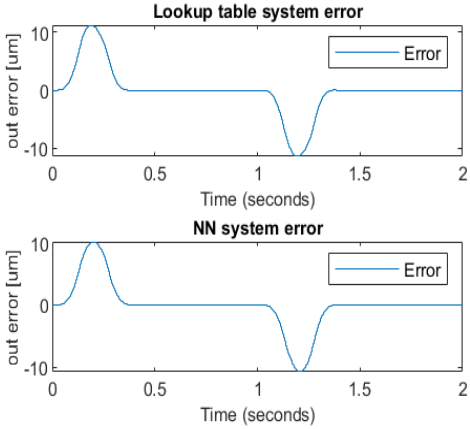

Fig. 10: Error with missing damping component included

In the future, what could be done to improve the performance of the overall system would be to estimate the inverse of the plant exactly. This will also capture the present nonlinear discrepancy in its entirety and will likely reduce the output error to a negligibly small value. To achieve this, a NN model can be implemented, which estimates the inverse transfer function of the plant and whose input depends only on the reference input position and the system output position. This will also be useful in the coming weeks, as these quantities are measurable and the model could be trained with data from the physical setup.

V. Conclusion

In this paper, the Simulink model of a high-precision positioning system and the principles of operation and training of neural network models were discussed. Furthermore, NN models were implemented in the Simulink model and tests were carried out with simulation data. It was observed that neural networks can approximate nonlinear relations very accurately and that they are a very effective method to improve system performance by accounting for nonlinear discrepancies.

Currently, the nonlinear discrepancy which causes the demonstrated system output error is not yet compensated. To do so, further research will need to directed towards a NN model, which predicts the inverse of the plant. The model predictions need to be based on inputs, which can be measured in the physical setup, so that experiments can be performed on the physical setup in the coming weeks

References

[1] J. Schoukens and L. Ljung, "Nonlinear system identification: a user–oriented road map," *Control systems magazine*, 2019.

[2] T. Nguyen, "Identification and compensation of parasitic effects in coreless linear motor," Ph.D. dissertation, Technische Universiteit Eindhoven, 2018.

[3] P. M. S. R. G. L. Le Ma, Patrick Bazzoli and D. A. Bristow, "Modeling and calibration of high-order joint-dependent kinematic errors for industrial robots," *Robotics and Computer Integrated Manufacturing*, 2018.

[4] M. S. Kurt Hornik and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, 1989.

[5] J. v. d. W. M. H. Johan Kon, Dennis Bruijnen and T. Oomen, "Physics-guided neural networks for feedforward control: An orthogonal projection-based approachs," *American Control Conference (ACC)4377-4382*, 2022.

[6] S. K. E. v. H. R. K. T. S.-B. N. S. Max Bolderman, Hans Butler and M. Lazar, "Physics-guided neural networks for feedforward control with input-to-state stability guarantee," *Control Engineering Practice*, 2024.
[7] M. L. Max Bolderman and H. Butler, "Physics–guided neural networks for inversion–based feedforward control applied to linear motors," *2022 IEEE 61st Conference on Decision and Control*, 2022.
[8] M. Bolderman, "Physics-guided neural networks for high-precision mechatronics: Applications to feedforward control and commutation," Ph.D. dissertation, Technische Universiteit Eindhoven, 2024.
[9] M. L. Max Bolderman and H. Butler, "Physics–guided neural networks for feedforward control: From consistent identification to feedforward controller design," *2022 IEEE 61st Conference on Decision and Control*, 2022.
[10] D. Fan, "Physics-guided neural networks for inversion-based feedforward control of a hybrid stepper motor," Ph.D. dissertation, Technische Universiteit Eindhoven, 2022.